Конспект лекции
Основы языка программирования Java

Цель и задачи лекции

Цель – ознакомиться с языком Java.

Задачи:

1. Изучить основы синтаксиса Java
2. Понять, как выполняется программа на языке Java
3. Дать определения понятиям переменная, объект, класс, метод
4. Изучить структуру объекта и его основные методы
5. Понять основы объектно-ориентированного программирования

План занятия

1. Отличия Java от других языков
2. Компиляция и исполнение Java-кода
3. Типы данных Java
4. Массивы
5. Понятие класса и объекта
6. Методы класса Object
7. Отношения между классами
8. Основы объектно-ориентированного программирования

Отличия Java от других языков

1. Java является одновременно и компилируемым и интерпретируемым языком (концепция байт-кода).
2. В Java реализована сборка мусора (будет рассмотрено на занятии «JVM, JIT, GC»).
3. Java свободно переопределяет порядок операций и очередность инструкций потоков по
своему усмотрению (но в рамках контрактов модели памяти; рассматривается в занятии
«Модель памяти Java»).
4. Java может свободно подменять некоторые инструкции в коде на более оптимальные
(будет рассмотрено на занятии «JVM, JIT, GC»).
5. В Java реализована JIT-компиляция (будет рассмотрено на занятии «JVM, JIT, GC»).

Java, это не просто еще один синтаксис для кода, это платформа.

Компиляция и исполнение Java-кода

Java-программа компилируется не в машинный язык, а в машинно-независимый код низкого уровня, байт-код. Далее байт-код выполняется виртуальной машиной.

Для выполнения байт-кода обычно используется интерпретация, хотя отдельные его части для ускорения работы программы могут быть транслированы в машинный код непосредственно во время выполнения программы по технологии компиляции «на лету» (Just-in-time compilation, JIT).

Для Java байт-код исполняется виртуальной машиной Java (Java Virtual Machine, JVM).

Подобный подход в некотором смысле позволяет использовать плюсы как интерпретаторов, так и

компиляторов.

Недостатки:

- необходима программа – интерпретатор
- медленнее компилируемых программ
- большие требования к ресурсам
- требование корректности исходного кода (при внесении изменений требуется перекомпиляция кода)

Достоинства:

- компактность
- независимость от ОС (переносимость)
- быстродействие

Типы данных Java

Java - строго типизированный язык. Именно этим объясняется безопасность и надежность программ на Java. Во-первых, каждая переменная и каждое выражение имеет конкретный тип, и каждый тип строго определен. Во-вторых, все операции присваивания, как явные, так и через параметры, передаваемые при вызове методов, проверяются на соответствие типов.

Целочисленные типы

Тип	**Размер (бит)**	**Диапазон**
byte	8 бит	от -128 до 127
short	16 бит	от -32768 до 32767

char	16 бит	беззнаковое целое число, представляющее собой символ UTF-16 (буквы и цифры)
int	32 бит	от -2147483648 до 2147483647
long	64 бит	от -9223372036854775808L до 9223372036854775807L

Над целочисленными аргументами можно производить следующие операции:

- операции сравнения (возвращают булево значение)
<, <=, >, >=, ==, !=
- числовые операции (возвращают числовое значение)
  - унарные операции + и -
  - арифметические операции +, -, *, /, %
  - операции инкремента и декремента (в префиксной и постфиксной форме): ++ и --
  - операции битового сдвига <<, >>, >>>
  - битовые операции ~, &, |, ^
- оператор с условием ?:
- оператор приведения типов
- оператор конкатенации со строкой +

Вычисления с целыми типами данных проводятся только с 32-х и 64-х битной точностью.

Типы с плавающей точкой

Тип	**Размер (бит)**	**Диапазон**
float	32	от -1.4e-45f до 3.4e+38f
double	64	от -4.9e-324 до 1.7e+308

Специальные значения:

- положительная и отрицательная бесконечности (positive/negative infinity);
- значение "не число", Not-a-Number, сокращенно NaN;
- положительный и отрицательный нули.

Логический тип

Тип	**Размер (бит)**	**Значение**
boolean	8 (в массивах), 32 (не в массивах используется int)	true (истина) или false (ложь)

Тип boolean предназначен для хранения логических значений и может принимать только одно из двух возможных значений: true или false. Данный тип всегда возвращается при использовании операторов сравнения (больше, меньше, равно, больше или равно, меньше или равно, не равно). Также он используется в управляющих операторах if и for.

Ссылочные типы и объекты

Ссылочные типы - это все остальные типы: классы, перечисления и интерфейсы, например, объявленные в стандартной библиотеке Java, а также массивы.

- Ссылочные переменные создаются с использованием определенных конструкторов

классов. Они предназначены для доступа к объектам. Эти переменные объявляются с

определенным типом, который не может быть изменен.

- Объекты класса и различные виды переменных массива подпадают под ссылочный тип

данных.

- По умолчанию в Java значение любой переменной ссылки - null.
- Ссылочная переменная может применяться для обозначения любого объекта,

объявленного или любого совместимого типа.

Классы обертки

Если требуется создать ссылку на один из примитивных типов данных, необходимо использовать соответствующий класс-обертку. Также в таких классах есть некоторые полезные методы и константы, например минимальное значение типа int можно узнать использовав константу Integer.MIN_VALUE. Оборачивание примитива в объект называется упаковкой (boxing), а обратный процесс распаковкой (unboxing).

Тип	**Класс-обертка**
byte	Byte
short	Short
int	Integer
long	Long
char	Character
float	Float
double	Double
boolean	Boolean

Рекомендуется использовать метод valueOf, поскольку он быстрее и использует меньше памяти потому за счет применения кэширования, а конструктор всегда создает новый объект.

Получить примитив из объекта-обертки можно методом <имя примитивного типа>Value.

Массивы

Массив - это группа однотипных переменных, для обращения к которым используется общее имя. В Java допускается создание массивов любого типа и разной размерности. Доступ к конкретному элементу массива осуществляется по его индексу. Массив предоставляют удобный способ группирования связанной вместе информации.

Одномерные массивы представляют собой список однотипных переменных. Чтобы создать, массив, нужно сначала объявить переменную массива требуемого типа. Общая форма объявления одномерного массива выглядит следующим образом:

```
тип[] имя_переменной;
```

где параметр тип обозначает тип элемента массива, называемый также базовым типом.

Многомерные массивы представляют собой массивы массивов.

При объявлении переменной многомерного массива для указания каждого дополнительного индекса используется отдельный ряд квадратных скобок. Например:

```
int twoD [][] = new int [4][5];
```

Понятие класса, объекта, метода

Класс - это элемент, составляющий основу Java. А поскольку касс определяет форму и сущность объекта, то он является той логической конструкцией, на основе которой построен весь язык java. Как таковой, класс образует основу объектно-ориентированного программирования на java. Особенность касса состоит в том, что он определяет новый тип данных.

Как только этот новый тип данных будет определен, им можно воспользоваться для создания объектов данного типа. Таким образом, класс - это шаблон для создания объекта, а объект – это экземпляр касса. А поскольку объект является экземпляром касса, то понятия объект и экземпляр употребляются как синонимы.

Для объявления класса служит ключевое слово class. Упрощенная общая форма определения класса имеет следующий вид:

```
class имя_класса {
    тип переменная_экземпляра1;
```

```
    тип переменная_экземпляра2;
    // ...
    тип переменная_экземпляраN;

    тип имя_метода 1 (список_параметров) {
        // тело метода
    }
    тип имя_метода2 (список_параметров) {
        // тело метода
    }
    // ...
    тип имя__методаN (список_параметров) {
        // тело метода
    }
}
```

Данные, или переменные, определенные внутри класса, называются переменными экземпляра. Код содержится внутри методов. Определенные внутри класса методы и переменные вместе называют членами класса. В большинстве классов действия с переменными экземпляров и доступ к ним выполняются через методы, определенные в этом классе. Таким образом, в общем случае именно методы определяют способ использования данных класса.

Члены класса

Класс может содержать члены (members) трех основных категорий.

Поля (fields) - переменные, относящиеся к классу и его объектам и в совокупности определяющие состояние класса или конкретного объекта.

Методы (methods) - именованные фрагменты исполняемого кода класса, обуславливающие особенности поведения объектов класса.

Вложенные типы - объявления классов или интерфейсов, размещенные в контексте объявлений других классов или интерфейсов.

Модификаторы объявления класса

1. Модификатор доступа public применим только к классам верхнего уровня и классам-членам, но не к локальным или анонимным классам.
2. Модификаторы доступа protected и private применимы только к классам членам в непосредственно охватывающем классе или к объявлениям перечислений.
3. Модификатор static применим только к классам-членам, но не к классам верхнего уровня, локальным или анонимным.
4. Класс может быть объявлен как final, если его определение завершено и никакие его подклассы в дальнейшем не могут потребоваться.
5. Воздействие модификатора strictfp состоит в том, что все float или double выражения в объявлении класса (включая выражения в инициализаторах переменных, инициализаторах экземпляров, статических инициализаторах и конструкторах) явным образом

становятся FP-строгими (используют переносимые механизмы для вычислений
с плавающей точкой)

Инициализация класса

Упрощенно любой класс можно описать в виде набора полей и методов.

```
public class MyClass {
    static int a = 1;
    static {
        a = 2;
    }

    int b = 3;
    {
        b = 4;
    }

    public MyClass {
        super();
    }

    public void foo() {}
}
```

Поля объекта инициализируются в следующем порядке:

1. Статические поля класса Parent;
2. Статический блок инициализации класса Parent;
3. Статические поля класса Child;
4. Статический блок инициализации класса Child;
5. Нестатические поля класса Parent;
6. Нестатический блок инициализации класса Parent;
7. Конструктор класса Parent;
8. Нестатические поля класса Child;
9. Нестатический блок инициализации класса Child;
10. Конструктор класса Child.

Важно, статические поля и блоки инициализируются в порядке их объявления в коде.

Порядок инициализации полей объекта:

- инициализация полей в месте объявления и в инициализационном блоке происходит до инициализации в конструкторе
- инициализации полей в месте объявления и в инициализационных блоках выполняются в порядке их объявления в классе
- инициализация полей базового класса происходит полностью до инициализации производного класса, т.е. сначала выполняются все инициализаторы базового класса, а потом все инициализаторы производного класса.

Интерфейсы

Объявление интерфейса вводит новый ссылочный тип, членами которого являются классы, интерфейсы, константы и методы. Этот тип не имеет переменных экземпляров и обычно объявляет один или несколько абстрактных методов; в противном случае несвязанные классы могут реализовывать интерфейс путем предоставления реализаций его абстрактных методов. Интерфейсы не могут быть инстанцированы непосредственно.

При этом класс может быть объявлен как непосредственно реализующий один или несколько интерфейсов. Каждый интерфейс неявно объявлен как abstract.

```
interface MyInterface {
    void method();
}
```

Если в объявлении имеется конструкция extends, то объявляемый интерфейс расширяет каждый из прочих именованных интерфейсов, а потому наследует типы-члены, методы и константы каждого из прочих именованных интерфейсов.

Любой класс, реализующий (т.е. содержащий конструкцию implements в объявлении) объявленный интерфейс, считается также реализующим интерфейсы, которые расширяют данный интерфейс.

```
class MyClass implements MyInterface {

    @Override
    public void method() {
        ...
    }
}
```

Состав объекта в Java

Для каждого объекта виртуальная машина JVM хранит:

1. Заголовок объекта
2. Память для примитивных типов
3. Память для ссылочных типов
4. Смещение/выравнивание

Каждый экземпляр класса содержит заголовок. Каждый заголовок для большинства JVM (Hotspot, OpenJVM) состоит из двух машинных слов.

	32-х разрядная система	**64-х разрядная система**
Размер заголовка	8 байт	16 байт

Методы класса Object

У класса есть несколько важных методов.

- Object clone() - создаёт новый объект, не отличающий от клонируемого
- boolean equals(Object obj) - определяет, равен ли один объект другому
- void finalize() - вызывается перед удалением неиспользуемого объекта
- Class<?> getClass() - получает класс объекта во время выполнения
- int hashCode() - возвращает хеш-код, связанный с вызывающим объектом
- void notify() - возобновляет выполнение потока, который ожидает вызывающего объекта
- void notifyAll() - возобновляет выполнение всех потоков, которые ожидают вызывающего объекта
- String toString() - возвращает строку, описывающий объект
- void wait() - ожидает другого потока выполнения
- void wait(long millis) - ожидает другого потока выполнения
- void wait(long millis, int nanos) - ожидает другого потока выполнения

Методы getClass(), notify(), notifyAll(), wait() являются финальными и их нельзя переопределять.

Метод clone()

Метод clone используется для создания дубликата объекта.

Метод equals()

Метод equals определяет понятие эквивалентности объектов, основанное на сравнении значений, а не ссылок.

Метод finalize()

Метод finalize выполняется непосредственно перед уничтожением объекта.

Иногда при уничтожении объект должен будет выполнять какое-либо действие. Например, если объект содержит какой-то ресурс, отличный от ресурса Java (вроде файлового дескриптора или шрифта), может требоваться гарантия освобождения этих ресурсов перед уничтожением объекта. Для подобных ситуаций Java предоставляет механизм, называемый финализацией. Используя финализацию, можно определить конкретные действия, которые будут выполняться непосредственно перед удалением объекта сборщиком мусора.

Чтобы добавить в класс средство выполнения финализации, достаточно определить метод finalize (). Среда времени выполнения Java вызывает этот метод непосредственно перед удалением объекта данного класса. Внутри метода finalize () нужно указать те действия, которые должны быть выполнены перед уничтожением объекта. Сборщик мусора запускается периодически, проверяя наличие объектов, на которые отсутствуют ссылки как со стороны какого-либо текущего состояния, так и косвенные ссылки

через другие ссылочные объекты. Непосредственно перед освобождением ресурсов среда времени выполнения Java вызывает метод finalize () по отношению к объекту.

Метод getClass()

Метод getClass возвращает объект типа Class, который представляет класс объекта.

Объект типа Class существует для каждого ссылочного типа. Он может использоваться, например, для выяснения полного квалифицированного имени класса, его членов, непосредственного суперкласса и реализуемых им интерфейсов

Метод hashCode()

Хеш-код – это целое число, генерируемое на основе конкретного объекта.

Этот метод реализован таким образом, что для одного и того-же входного объекта, хеш-код всегда будет одинаковым. Следует понимать, что множество возможных хеш-кодов ограничено примитивным типом int. Из-за этого ограничения, вполне возможна ситуация, что хеш-коды разных объектов могут совпасть. Следует участь, что:

1) Если хеш-коды разные, то и входные объекты гарантированно разные.

2) Если хеш-коды равны, то входные объекты не всегда равны.

3) Если объекты равны, то хеш-коды равны.

Ситуация, когда у разных объектов одинаковые хеш-коды называется — коллизией.

Вероятность возникновения коллизии зависит от используемого алгоритма генерации хешкод.

Методы wait(), notify() и notifyAll()

Методы wait, notify и notifyAll используются в параллельном программировании с использованием потоков.

Метод wait() освобождает монитор и переводит вызывающий поток в состояние ожидания до тех пор, пока другой поток не вызовет метод notify().

Метод notify() продолжает работу потока, у которого ранее был вызван метод wait().

Метод notifyAll() возобновляет работу всех потоков, у которых ранее был вызван метод wait().

Все эти методы вызываются только из синхронизированного контекста -- синхронизированного блока или метода.

- public final void wait(long timeout, int nanos) throws InterruptedException

- public final void wait() throws InterruptedException

По прошествии времени, переданного в аргументах - пытается захватить монитор и продолжить выполнение программы.

Методы toString()

Метод toString возвращает представление объекта в виде строки String.

Отношения между классами

Агрегация — отношение, когда один объект является частью другого. Например, Студент входит в Группу любителей физики.

Композиция — еще более «жесткое отношение, когда объект не только является частью другого объекта, но и вообще не может принадлежат еще кому-то. Например, машина и двигатель. Хотя двигатель может быть и без машины, но он вряд ли сможет быть в двух или трех машинах одновременно. В отличии от студента, который может входить и в другие группы тоже.

Ассоциация – отношение между классами объектов, которое позволяет одному экземпляру объекта вызвать другого, чтобы выполнить действие от его имени. Это структурное отношение, поскольку определяет связь между объектами одного рода и объектами другого рода и не моделирует поведение.

Наследование – это процесс, посредством которого один объект может приобретать свойства другого. Точнее, объект может наследовать основные свойства другого объекта и добавлять к ним черты, характерные только для него.

Основы объектно-ориентированного программирования

Полиморфизм

Полиморфизм — это возможность использования в одном и том же контексте различных программных сущностей (объектов, типов данных и т. д.) с одинаковым интерфейсом. Полиморфизм позволяет объектам принимать несколько различных форм.

В Java доступны следующие виды полиморфизма:

1. перегрузка методов (ad-hoc-полиморфизм):

```
public static String toString(long value) { ... }
public static String toString(double value) { ... }
```

2. полиморфизм подтипов — наследование и иерархия классов:

```
public class Base {
    public void method() { ... }
```

```
}
public class Child extends Base {
    @Override public void method() { ... }
}
public class Child extends Base {
    @Override public void method() { ... }
}
```

3. параметрический полиморфизм — дженерики и параметризованные классы:

```
public class List<T> {
    public T get(int index) { ... }
}
public static <T> T getInstance(Class<T> clazz) {
   ...
}
```

Наследование

Наследование — принцип ООП, позволяющий описать новый класс на основе уже существующего (родительского), при этом свойства и функциональность родительского класса заимствуются новым классом.

```
public class A {
    private int x = 1;

    public void foo() {
        System.out.println("A method");
    }
}

public class B extends A {
    private int x = 1;

    @Override
    public void foo() {
        System.out.println("B method");
    }
}
```

Инкапсуляция

Инкапсуляция – это свойство системы, позволяющее объединить данные и методы, работающие с ними, в классе и скрыть детали реализации от пользователя.

```
public class A {
    public int x = 1;
    private int y = 2;

    public void go() {
        System.out.println("method Go");
    }

    private void stop() {
        System.out.println("method Stop");
    }
}
```

```
public class B {
    public void methodB() {
        A a = new A();
        int varX = a.x;
        int varA = a.y; //ошибка доступа, т.к. переменная с модификатором private
        a.go();
        a.stop();    //ошибка доступа, т.к. метод приватный
    }
}
```

Абстракция

Абстракция – это набор всех характеристик, которые позволяют выделить значимые характеристики объекта, исключая из рассмотрения незначимые. Основная идея состоит в том, чтобы представить объект минимальным набором полей и методов и при этом с достаточной точностью для решаемой задачи.

```
class People {
    int age;
    Sex sex;

    void live() {
        ...
    }

    void work() {
        ...
    }

    void eat() {
        ...
    }
}
```

Литература и ссылки

1. Спецификация языка Java https://docs.oracle.com/javase/specs/jls/se8/html/index.html
2. Спецификация JVM https://docs.oracle.com/javase/specs/jvms/se8/jvms8.pdf
3. Курс по Java на intuit http://www.intuit.ru/studies/courses/16/16/info
4. лекции 1-9
5. Доклад "Боремся за память в Java" http://jug.ua/wp-content/uploads/2013/02/Java.pdf
6. Java code conventions http://www.oracle.com/technetwork/java/codeconventions-150003.pdf

Вопросы для самоконтроля

1. Что такое байт-код?
2. Насколько отличается размер объекта int и Integer в 32-битной системе?
3. Какое значение по умолчанию возвращает toString? hashCode?

4. От каких объектов можно вызвать метод clone()?
5. В чем отличие между агрегацией и композицией?